



KATHOLIEKE  
UNIVERSITEIT  
LEUVEN

# **DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN**

RESEARCH REPORT 0028

**MAXIMIZING THE NET PRESENT VALUE OF A  
PROJECT WITH PROGRESS PAYMENTS**

by

**M. VANHOUCKE  
E. DEMEULEMEESTER  
W. HERROELEN**

D/2000/2376/28

# **MAXIMIZING THE NET PRESENT VALUE OF A PROJECT WITH PROGRESS PAYMENTS**

**Mario VANHOUCKE • Erik DEMEULEMEESTER • Willy HERROELEN**

**June 2000**

Operations Management Group  
Department of Applied Economics  
Katholieke Universiteit Leuven  
Naamsestraat 69, B-3000 Leuven (Belgium)  
Phones: 32-16-32 69 65, 32-16-32 69 72, 32-16-32 69 70, Fax 32-16-32 67 32  
E-mail: <first name>.<name>@econ.kuleuven.ac.be

# MAXIMIZING THE NET PRESENT VALUE OF A PROJECT WITH PROGRESS PAYMENTS

**Mario Vanhoucke**

mario.vanhoucke@econ.kuleuven.ac.be

**Erik Demeulemeester**

erik.demeulemeester@econ.kuleuven.ac.be

**Willy Herroelen**

willy.herroelen@econ.kuleuven.ac.be

*Operations Management Group*

*Department of Applied Economics, Katholieke Universiteit Leuven*

*Naamsestraat 69, B-3000 Leuven (Belgium)*

## ABSTRACT

In this paper we study the unconstrained project scheduling problem with discounted cash flows where the net cash flows are assumed to be dependent on the completion times of the corresponding activities. Cash outflows occur when an activity is completed whereas cash inflows are incurred as progress payments at the end of some time period. The objective is to schedule the activities in order to maximize the net present value (*npv*) of the project subject to the precedence constraints and a fixed deadline. This paper extends the ever growing amount of research concerning the financial aspects in project scheduling in which cash flows are time-dependent.

We introduce a branch-and-bound algorithm which computes upper bounds by making piecewise linear overestimations. In doing so, the algorithm transforms the problem into a weighted earliness-tardiness project scheduling problem. The algorithm is extended with two new rules in order to reduce the size of the branch-and-bound tree. Computational results are reported since the procedure has been coded in Visual C++ version 4.0 under Windows NT and has been validated on a randomly generated problem set.

**Keywords:** *Project scheduling; Net present value; Progress payments; Optimal search*

## 1. Introduction

Since the introduction of cash flows in project scheduling problems by Russell (1970), the maximization of the net present value has gained increasing attention throughout the literature. This has led to a large amount of algorithms presented by Russell (1970), Grinold (1972), Elmaghraby and Herroelen (1990), Herroelen and Gallens (1993), Doersch and Patterson (1977), Russell (1986), Smith-Daniels and Aquilano (1987), Smith-Daniels and Smith-Daniels (1987), Patterson et al. (1989, 1990), Baroum (1992), Yang et al. (1992, 1995), Padman and Smith-Daniels (1993), Icmeli and Erengüç (1994, 1996), Özdamar et al. (1994), Ulusoy and Özdamar (1995), Baroum and Patterson (1996, 1999), Pinder and Marucheck (1996), Smith-Daniels et al. (1996), Zhu and Padman (1996, 1997), Padman et al. (1997) and Vanhoucke et al. (1999a). In these problems cash flows occur throughout the life of the project according to a rich variety of possible patterns (positive and/or negative, event-oriented or activity-based). One of the main characteristics of this previous research is the independency of the cash flows with respect to the completion times of the activities.

Since in a large number of projects the cash flows depend on their occurrence in different ways, recent research has focused on the case where activity cash flows are dependent on the completion times of the corresponding activities. To the best of our knowledge, the research efforts concerning this type of cash flows are done by Dayanand and Padman (1993a, 1993b, 1997), Etgar et al. (1996), Kazaz and Sepil (1996), Sepil and Ortaç (1997), Shtub and Etgar (1997), Etgar and Shtub (1999) and Vanhoucke et al. (1999b). For a comprehensive review concerning the different types of time-interdependencies, we refer to Vanhoucke et al. (1999b).

In this paper we present an exact branch-and-bound procedure to maximize the net present value in activity-on-the-node project networks with zero-lag finish-start precedence constraints where the activity-based cash flows are dependent on the completion times of the corresponding activities as follows: the cash outflows occur when an activity is completed whereas cash inflows are incurred as progress payment at the end of some time period for the work completed during this fixed time period (problem  $cpm, \delta_n, cf^{pp} | npv$ , following the classification scheme of Herroelen et al. (1999) and further denoted as the  $max-npv^{pp}$  problem). To the best of our knowledge, the problem has only been solved by Kazaz and Sepil (1996) for activity-on-the-arc networks. In their paper a mixed integer formulation of the problem is presented. They have tested their formulation on a randomly generated problem set using LINDO. Their results have revealed that the Benders Decomposition technique is the most promising one. A year later, Sepil and Ortaç (1997) have extended this problem by introducing renewable resources. They have developed three heuristic rules to solve the problem.

The organization of the paper is as follows. In section 2 we discuss the  $max-npv^{pp}$  problem. In section 3 we describe some features of this problem. Section 4 describes the logic of the exact procedure and deals with the introduction of two rules. In section 5 we illustrate the algorithm by means of an example. Section 6 tests its performance and reports detailed computational results. Section 7 gives an overall conclusion.

## 2. The max-npv<sup>pp</sup> problem

The deterministic max-npv<sup>pp</sup> problem involves the scheduling of project activities within a certain deadline  $\delta_n$  in order to maximize the net present value (npv) of the project in the absence of resource constraints. The project is represented by an activity-on-the-node (AoN) network where the set of nodes,  $N$ , represents activities and the set of arcs,  $A$ , represents finish-start precedence constraints with a time lag of zero. The activities are numbered from the dummy start activity 1 to the dummy end activity  $n$ . The duration of an activity is denoted by  $d_i$  ( $1 \leq i \leq n$ ) and the performance of each activity involves a series of cash flow payments and receipts as follows: we assume each activity  $i$  has a cash outflow  $c_i^-$  and a profit margin  $\gamma_i$ . The revenue from this activity  $i$  then amounts to  $c_i^+ = c_i^- (1 + \gamma_i)$ . We further assume that the cash outflows  $c_i^-$  occur at the completion of the activities, which is often the case when activities are subcontracted and the subcontractors are paid when the activities are completed. It is also possible that the cash outflows occur during the performance of the activity. When this is the case,  $c_i^-$  denotes the terminal value of these cash outflows, i.e. the compounded value of the associated cash outflows towards the completion of the activity. The cash inflows  $c_i^+$  are incurred as progress payments at the end of some time period. These progress payments are received at fixed points in time for the work completed during the current period, i.e. for the finished and partially finished activities.

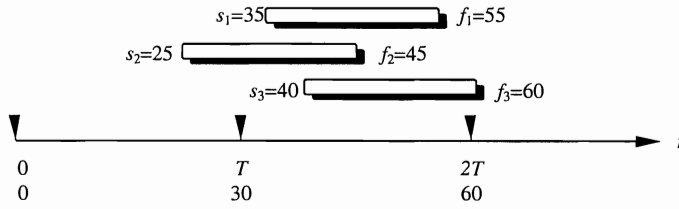
When  $\alpha$  represents the discount rate and the nonnegative variable  $f_i$  denotes the completion time of activity  $i$ , the discounted value of the cash outflows at the beginning of the project equals  $c_i^- e^{-\alpha f_i}$ . Although these cash outflows occur at the completion of the activities, this is not the case with the cash inflows. The cash inflows are incurred at the end of some time period  $T$  (e.g.  $T$  equals a month). This means that at the end of each period (i.e. at time instants  $T, 2T, \dots, mT$ , with  $mT \geq \delta_n$  and  $(m-1)T < \delta_n$ ). Remark that, when  $mT > \delta_n$ , the last period equals  $\delta_n$  the progress payments are received for the work completed at this time period  $T$ . When  $w_{it}$  ( $1 \leq i \leq n$  and  $t = T, 2T, \dots, mT$ ) denotes the portion of work completed during period  $]t-T, t]$  for activity  $i$  then the discounted value of the cash inflows at the beginning of the project for activity  $i$  equals  $\sum_{t=T, 2T, \dots, mT} w_{it} c_i^+ e^{-\alpha t}$ .

Consider Fig. 1 which displays the scheduling of three activities in time as described by Kazaz and Sepil (1996). The nonnegative integer variable  $s_i$  denotes the start time of activity  $i$ , whereas the completion time of this activity is denoted by  $f_i$ . For ease of explanation, we assume that all the activities have a duration of 20, i.e.  $d_1 = d_2 = d_3 = 20$ . Assume the cash outflow for each activity is  $c_1^- = c_2^- = c_3^- = 600$  and the profit margin equals  $\gamma_1 = \gamma_2 = \gamma_3 = 0.20$  (20%). Consequently, the revenue of each activity amounts to  $c_1^+ = c_2^+ = c_3^+ = 600(1+0.20) = 720$ . When  $\alpha$  equals 0.00167, the discounted value of the cash flows can be calculated as follows:

**activity 1:** the discounted value of the cash outflow equals  $600 e^{-\alpha 55} = 547.35$ . The execution of this activity lies completely in the second period  $[T, 2T]$  and therefore, the discounted value of the cash inflow equals  $720 e^{-\alpha 60} = 651.35$ . The net cash flow of activity 1 amounts to 104.00.

**activity 2:** At the end of the first time period (at time instant  $T$ ), 25% of activity 2 has already completed, i.e.  $w_{2,T} = 5/20$ . The remaining work is executed during the second time period and therefore  $w_{2,2T} = 15/20$ . The discounted value of the cash inflow at the beginning of the project amounts to  $w_{2,T} 720 e^{-\alpha 30} + w_{2,2T} 720 e^{-\alpha 60} = 171.20 + 488.51 = 659.71$ . Subtracting the discounted value of the cash outflow  $600 e^{-\alpha 45}$  results in a net discounted value of 103.16.

**activity 3:** Both the cash outflows and the cash inflows are incurred at the same time, i.e. at time instant 60. Consequently, the net discounted value of the cash flows amounts to  $(720 - 600) e^{-\alpha 60} = 108.56$ .



**Fig. 1.** The scheduling of three activities

A conceptual formulation of the  $\max\text{-}npv^{pp}$  problem can be given as follows:

$$\text{Maximize } \sum_{i=1}^n (-c_i^- e^{-\alpha f_i} + \sum_{t=T, 2T, \dots, mT} w_{it} c_i^+ e^{-\alpha t}) \quad [1]$$

Subject to

$$f_i \leq f_j - d_j \quad \forall (i, j) \in A \quad [2]$$

$$w_{it} = \frac{t - f_i + d_i}{d_i}, \text{ if } f_i - d_i < t \leq f_i \quad [3]$$

$$w_{it} = \min\{1, \frac{f_i + T - t}{d_i}\}, \text{ if } t - T < f_i < t$$

$$w_{it} = 0, \text{ otherwise}$$

$$i = 1, \dots, n \quad \text{and} \quad t = T, 2T, \dots, mT$$

$$f_n \leq \delta_n \quad [4]$$

$$f_1 = 0 \quad [5]$$

The objective in Eq. 1 maximizes the net present value of the project. The constraint set in Eq. 2 maintains the finish-start precedence relations among the activities. Eq. 3 represents the portion of work completed during period  $]t-T, t]$  for each activity  $i$ . This equation assumes, for ease of representation, that  $d_i \leq T$ . In order to restrict the project duration, we add a negotiated project deadline  $\delta_n$  given in Eq. 4. Eq. 5 forces the dummy start activity to end at time zero.

In the next section we discuss the exact solution procedure for the max- $npv^{pp}$  problem as formulated above in Eqs. [1] - [5].

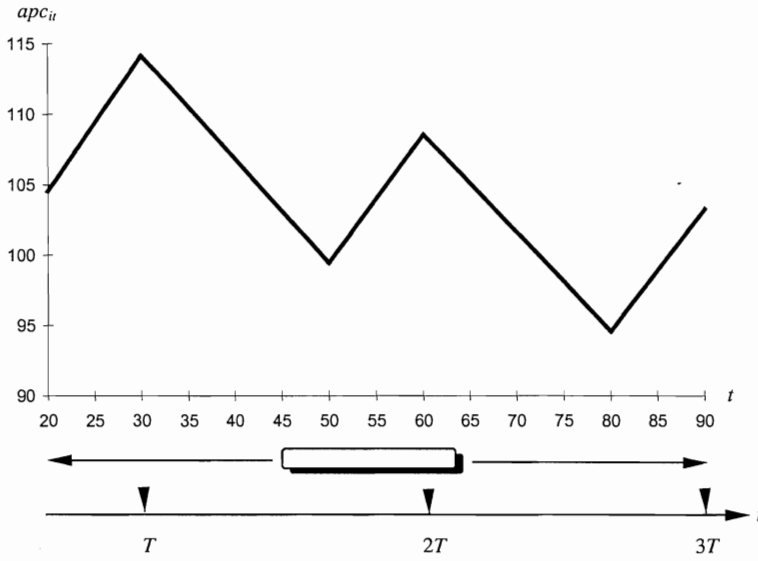
### 3. The max- $npv^{pp}$ features

In their paper, Kazaz and Sepil (1996) define the so-called *activity profit curve* as a graph which shows that the net present value of cash flows associated with an activity changes with respect to the activity completion times. Since the activity completion times vary between the critical path based earliest and the deadline based latest finishing time, the activity profit curve  $apc_{it}$  simply gives a graphical representation of the net present value of the cash flows of activity  $i$  over time  $t$ .

According to the authors, the activity profit curves satisfy the following conditions:

- the daily discount rate  $\alpha$  is the main parameter that affects the overall shape of the profit curves,
- the duration of the activity affects the time points where the curve changes the slope,
- the profit margin  $\gamma_i$  and the cost  $c_i$  of the activity mainly affect the steepness of the line segments in each interval.

More precisely, they have shown that the activity profit curve is a non-linear function of the activity finishing time and the discount rate. However, when the per period discount rate  $\alpha$  is smaller than 0.02, the function can be approximated by a linear one. The authors argue that this is a very realistic assumption since a daily discount rate  $\alpha$  greater than 0.02 corresponds to an annual discount rate of 3070%, a very unrealistic value. In the sequel of this paper we assume discount rates lower than 0.02. Therefore, we can make use of piecewise linear activity profit curves as shown in Fig. 2. This figure shows the net present value for the different finishing times of an activity taken from Fig. 1. Notice that the activity profit curve is piecewise linear with local maxima at time instants  $T, 2T, 3T, \dots, mT$ . Remark that  $d_i = 20$ ,  $\alpha = 0.00167$ ,  $c_i = 600$ ,  $\gamma_i = 0.2$ ,  $\delta_n = 90$  and  $T = 30$ . The earliest finishing time  $ef_i$  equals 20 while the latest finishing time  $lf_i$  equals 90.



**Fig. 2.** A piecewise linear activity profit curve  $apc_{it}$

It is easy to show that an activity profit curve, as shown in Fig. 2, consists of a number of weighted earliness-tardiness trade-offs. To that end, consider Fig. 3 in which the activity profit curve is subdivided in three different weighted earliness-tardiness trade-offs. Let  $h_i$  be the due date of the activity and  $e_i$  and  $t_i$  the earliness and tardiness cost, respectively. For each interval, the due date  $h_i$  corresponds to a point in time with a maximal net present value while both the earliness cost  $e_i$  and the tardiness cost  $t_i$  represent the slope of the line segments in each interval. In the *weighted earliness-tardiness project scheduling problem (WETPSP)*, each activity has a certain due date  $h_i$  and an earliness cost  $e_i$  as well as a tardiness cost  $t_i$ . The *WETPSP* involves the scheduling of project activities in order to minimize the weighted earliness-tardiness costs in the absence of resource constraints. Maximizing the net present value for each of the three intervals as shown in Fig. 3 is similar to minimizing the weighted earliness-tardiness costs.



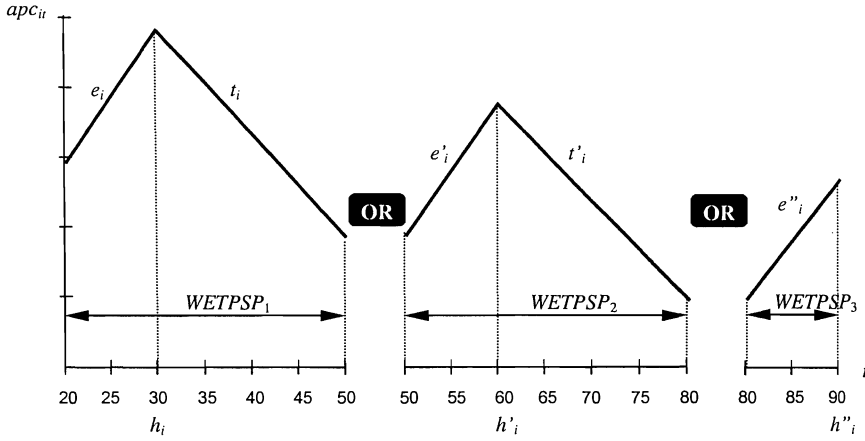


Fig. 3. Weighted earliness-tardiness trade-offs in an activity profit curve

In the next section, we describe the exact branch-and-bound algorithm for the  $\max\text{-}npv^{pp}$  problem. The procedure presented in this section provides an exact solution to the problem using an upper bound calculation by means of the weighted earliness-tardiness concept as described above.

#### 4. The exact algorithm

In this section we provide a branch-and-bound procedure for the  $\max\text{-}npv^{pp}$  problem using an adapted version of the recursive procedure by Vanhoucke et al. (2000) for the WETPSP for calculating the upper bounds.

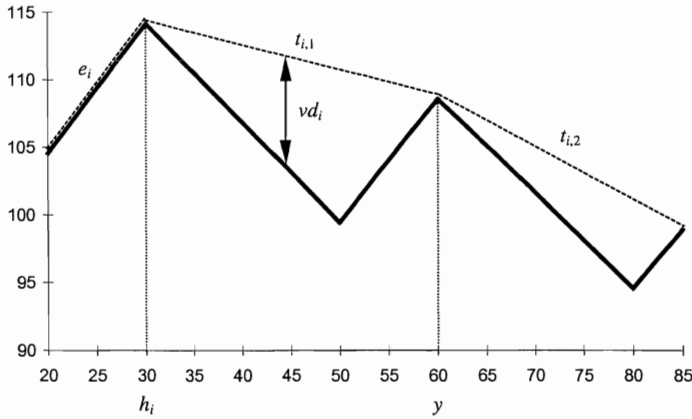
##### 4.1 Computing an upper bound

For calculating an upper bound for the  $\max\text{-}npv^{pp}$  problem we compute a piecewise linear overestimation curve of each activity's profit curve as shown in dotted lines in Fig. 4. Therefore, we assign a due date  $h_i$  to each activity which corresponds to a point in time with a maximal net present value. Earliness costs  $e_i$  and tardiness costs  $t_i$  are used to denote the slope of the linear overestimation curve. Remark that the tardiness cost of Fig. 4. is subdivided in two values  $t_{i,1}$  and  $t_{i,2}$  corresponding to the two different slopes at the right side of the due date  $h_i$ . In doing so, the problem has been transformed into a WETPSP and can be solved by an adapted version of the recursive procedure of Vanhoucke et al. (2000). In this two-step procedure, a due date tree is calculated in which all activities are scheduled at their due date or later. In a second step a recursive search identifies sets of activities which can be shifted backward (towards time zero) in order to decrease the weighted earliness-tardiness costs (and consequently, increase the net present value since minimizing the weighted earliness-tardiness costs corresponds to maximizing the net present value). Once a set of activities  $SA$  has been identified which can be shifted backward, the algorithm calculates an allowable displacement interval: the set of activities  $SA$  may be shifted until an activity  $i \in SA$  connects with either a predecessor

activity  $j \notin SA$  or with its due date  $h_i$ . For the computation of the upper bound, two additional criteria are used for the computation of the displacement interval. An activity  $i \in SA$  can be shifted backward in time until it reaches a breakpoint in the piecewise linear overestimation curve (as denoted by  $y$  in Fig. 4) and the set  $SA$  can only be shifted backward in time until one of its activities  $i$  reaches its earliest finishing time  $ef_i$ . Consequently, the allowable displacement interval will be calculated as  $\min \{v_{kl}, w, w', w''\}$ , with  $v_{kl} = \min_{\substack{(k,l) \in A \\ k \notin SA \\ l \in SA}} \{f_l - d_l - f_k\}$  and  $w = \min_{\substack{i \in SA \\ f_i > h_i}} \{f_i - h_i\}$ ,

according to the original definition in Vanhoucke et al. (2000) and with  $w'$  the difference between the finishing time and the breakpoint  $y$ , i.e.  $w' = \min_{\substack{i \in SA \\ f_i > y}} \{f_i - y\}$  and

$w''$  the difference between the finishing time and the earliest finishing time, i.e.  $w'' = \min_{i \in SA} \{f_i - ef_i\}$ .



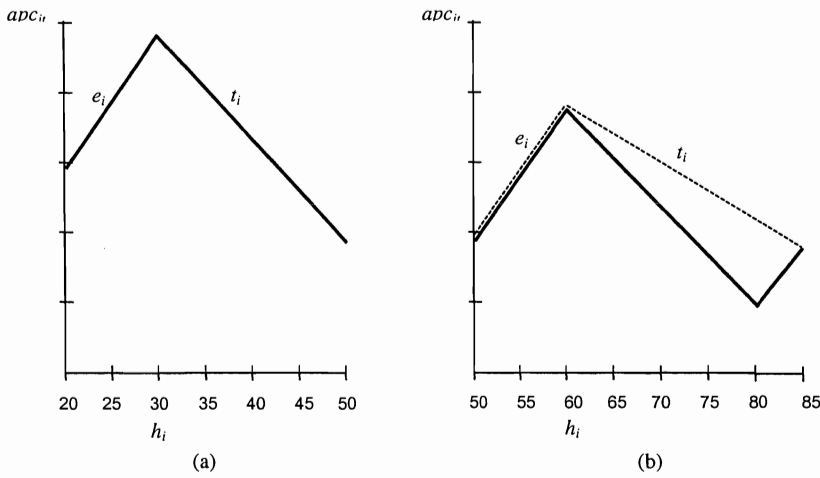
**Fig. 4.** An overestimation for an activity's profit curve

## 4.2 The branch-and-bound algorithm

An initial upper bound is obtained by solving the *WETPSP* as described in section 4.1. The algorithm reports the finishing times  $f_i$  of the activities with their corresponding net present value. For each activity, a vertical distance  $v d_i$  is computed which measures the quality of the piecewise linear overestimation. The vertical distance simply denotes the distance between the overestimation curve and the real net present value as shown in Fig. 4.

If no activity can be found for which  $v d_i > 0$ , then the found solution is exact and no branching is needed. Otherwise, branching is done by identifying the activity with the largest vertical distance (the so-called branching activity  $ba$ ) and partitioning the activity profit curves into two disjoint subsets. Therefore, we search for the minimum point  $t$  of the interval  $[xT, (x+1)T]$  for which  $f_i \in [xT, (x+1)T]$ , with  $x$  a positive integer number ( $t$  equals 50 in Fig. 4). The first subset (a) contains the

activity's profit curve to the left of this point  $t$  while the second subset (b) represents the activity's profit curve to the right of point  $t$ . These subsets are used to obtain two new piecewise linear overestimation curves, as displayed in Fig. 5. Solving these two new problems by means of the adapted *WETPSP* procedure yields the new corresponding upper bounds. Branching continues from the node with the largest upper bound. If in the new found solution no activity can be found with  $vd_i > 0$ , then we update the lower bound  $lb$  of the project (initially set to  $-\infty$ ) and explore the second node at that level of the branch-and-bound tree. Backtracking occurs when the calculated upper bound is smaller than or equal to the lower bound  $lb$ . The algorithm stops when we backtrack to the initial level of the branch-and-bound tree.



**Fig. 5.** Partitioning the activity profit curve into two disjoint subsets

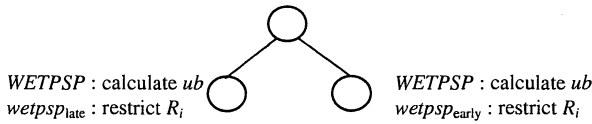
Remark that a number of additional steps need to be performed while partitioning the activity profit curves into two disjoint subsets. In the left branching node (a), the latest finishing time  $lf_{ba}$  of the branching activity  $ba$  is updated and a new overestimation curve will be calculated. Consequently, the latest finishing times of the predecessors  $j$  (either direct or transitive) of the branching activity  $ba$  must be updated too. Moreover, the overestimation curves of these predecessors  $j \in \bar{P}_{ba}$  must be updated in order to improve the upper bound calculation. A similar reasoning can be made for the right branching node (b) where the earliest finishing times  $ef_{ba}$  and the piecewise linear overestimation curves of the branching activity  $ba$  and its successors  $j \in \bar{S}_{ba}$  (either direct or transitive) have to be updated. If, due to these updates, the latest finishing time of an activity becomes smaller than its earliest finishing time, then the problem is infeasible and this node of the branch-and-bound tree will be fathomed.

### 4.3 Reducing the number of nodes in the branch-and-bound tree

In order to reduce the number of nodes in the branch-and-bound tree, we extend the branch-and-bound algorithm with two new rules. The first rule reduces the range  $R_i$  of each activity  $i$  while the second rule calculates an alternative upper bound for the nodes in the tree.

**Rule 1:** *Reducing the range  $R_i$  of each activity  $i$*

During the branch-and-bound procedure, the piecewise linear overestimation curves as well as the earliest and latest finishing times of a number of activities are modified along the exploration of the tree. Remark that the branch-and-bound procedure constructs a binary tree in which at each level the profit curve of the branching activity is partitioned into two disjoint subsets as shown in Fig. 6. At the left node, only the latest finishing times of the branching activity and its predecessors (both immediate and transitive) are updated in comparison with its parent node. For the exploration of the right node, the opposite is true: only the earliest finishing times of the branching activity and its successors are updated in comparison with its parent node.



**Fig. 6.** A binary tree: each parent node generates two child nodes

In order to reduce the number of nodes in the branch-and-bound tree, it is useful to restrict the range  $R_i$ , defined as the interval between  $ef_i$  and  $lf_i$ , of each activity  $i$ , resulting in a better overestimation curve. To that purpose we call the recursive search procedure for the *WETPSP* twice at each node: once for the upper bound calculation and once to restrict the range of the activities. The second call (referred to as *wetpspearly* or *wetpsplate* in Fig. 6, dependent on which node the procedure is performed) leads to a considerable reduction of the number of nodes in the branch-and-bound tree and is described in the sequel of this section.

Each activity's profit curve can be subdivided in a number of weighted earliness-tardiness trade-offs, as shown for an example activity  $i$  in Fig. 7(a). During the calculation of the upper bound by means of the recursive *WETPSP* procedure, each activity is overestimated as displayed in Fig. 7(b). Now suppose that, before starting the upper bound calculation, we solve the *WETPSP* problem with each activity restricted to its first earliness-tardiness trade-off interval, i.e. interval *WETPSP*<sub>1</sub> of Fig. 7(a) (referred to as *wetpspearly*-procedure). The solution obtained can be used to update the earliest finishing times of each activity and consequently to modify the overestimation curves based on the following observation:

*The finishing times  $f'_i \mid i \in N$  reported by the  $wetpsp_{\text{early}}$ -procedure (i.e. by restricting the range  $R_i$  of each activity to its first interval) are smaller than or equal to the finishing times  $f_i \mid i \in N$  when the activity ranges are not restricted at all.*

*Proof:*

It is sufficient to show that it is impossible to find an activity  $i$  which is scheduled at its finishing time  $f_i < f'_i$  by means of the  $WETPSP$ -procedure (in which the ranges of the activities are not restricted at all).

Suppose that the opposite is true, i.e. at least one activity can be found for which its completion time reported by the  $WETPSP$ -procedure is smaller than its completion time reported by the  $wetpsp_{\text{early}}$ -procedure, i.e.  $\exists i \in N \mid f_i < f'_i$ . This, however, can only happen due to the fact that activity  $i$  is forced to finish earlier than  $f'_i$  by one or more successors (otherwise, the activity would have been scheduled at its finishing time  $f'_i$  or later). Denote this successor activity by  $j$  (the case when more than one successor is responsible for the scheduling of activity  $i$  is analogous) and consequently the binding precedence relation between activity  $i$  and  $j$  is represented by  $f_i + d_j = f_j$ .

Now we have  $f_i + d_j = f_j$ , as a result from the  $WETPSP$ -procedure, and  $f'_i + d_j \leq f'_j$ , as a result from the  $wetpsp_{\text{early}}$ -procedure. Since we have assumed that  $f_i < f'_i$  it follows that  $f_j < f'_j$ . Since  $f'_j$  denotes the finishing time of activity  $j$  when its range is restricted to the first interval, it follows that the finishing time of activity  $j$  reported by the  $WETPSP$ -procedure (i.e.  $f_j$ ) falls in its first interval. This is, however, considered by the  $wetpsp_{\text{early}}$ -procedure in which the optimal finishing times  $f'_i$  ( $f'_i > f_i$ ) and  $f'_j$  were reported. Consequently, it is impossible to find an activity for which  $f_i < f'_i$  and therefore  $f_i \geq f'_i$ .

Since the finishing time  $f'_i$  reported by the  $wetpsp_{\text{early}}$ -procedure is smaller than or equal to the finishing time  $f_i$  when the activity ranges are not restricted at all, we can restrict the range of activity  $i$  by replacing its earliest finishing time  $e_i$  by  $f'_i$ .

This results in a better overestimation curve as shown in Fig. 7.

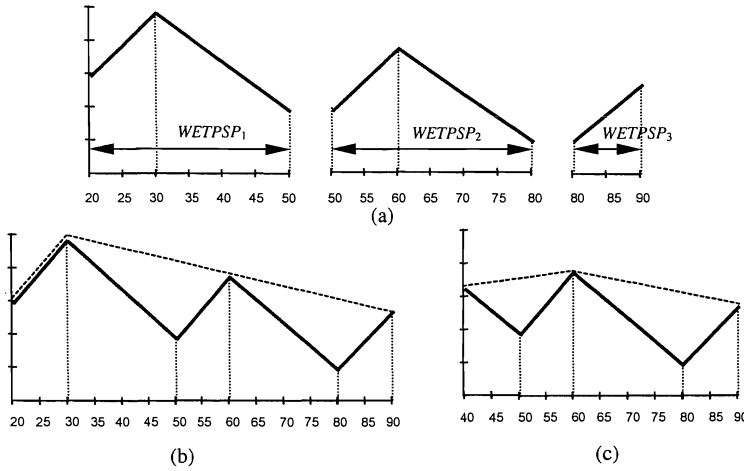
*Q.E.D.*

The same logic can be applied to update the latest finishing times of the activities. In this case, the recursive  $WETPSP$  procedure will be called with each activity in its last earliness-tardiness trade-off interval, i.e.  $WETPSP_3$  in Fig. 7(a) (referred to as  $wetpsp_{\text{late}}$ -procedure). The finishing times  $f'_i \mid i \in N$  reported by this procedure are always greater than or equal to the finishing times  $f_i \mid i \in N$  when the activity ranges are not restricted at all.

Recall that at the left node, only the latest finishing times of the branching activity and its predecessors are updated in comparison with its parent node. Consequently, there is no need to solve the problem with the  $wetpsp_{\text{early}}$ -procedure since the earliest finishing times of the activities have remained unchanged. It is, however, advantageous to solve the problem with the  $wetpsp_{\text{late}}$ -procedure in order to

update certain latest finishing times. For the exploration of the right node, the opposite is true. Only the  $wetpsp_{\text{early}}$ -procedure will be called in order to restrict the range  $R_i$  of each activity  $i$  by updating certain earliest finishing times.

Consider the following example: assume that the  $wetpsp_{\text{early}}$ -procedure reports a finishing time for activity  $i$  that amounts to  $f'_i=40$  (given that each activity's profit curve is restricted to the first earliness-tardiness trade-off, e.g.  $ef_i = 20$  and  $lf_i = 50$  in Fig. 7). This means that the finishing time  $f_i$  of activity  $i$  will be greater than or equal to 40 when its range  $R_i$  equals  $(20,90)$  (and, consequently, it is not restricted to the first interval). Therefore, we replace the earliest finishing time of this activity time  $ef_i = 20$  by its reported finishing time  $f'_i = 40$ . That is  $ef_i = 40$ , as shown in Fig. 7(c). This results in a much better overestimation curve. This restriction can be applied to all activities  $i \in N$ .



**Fig. 7.** Overestimation curve with (c) and without (b) restricting the range  $R_i$

## Rule 2: Calculating an alternative upper bound

In what follows, we use  $cn$  to denote the current node at level  $p$  which is already fully explored and  $nn$  the new node that we want to explore at this level, i.e. if  $cn$  refers to the right branching node than  $nn$  refers to the left branching node as displayed in Fig. 8 or vice versa. Furthermore, let  $ln$  denote the best leaf node (i.e. the leaf node with the highest  $npv$ ) from the subtree rooted at the current node  $cn$  and  $ub_{cn}$  and  $ub_{nn}$  the upper bound calculated by the  $WETPSP$  procedure for node  $cn$  and  $nn$  at level  $p$ , respectively. Branching occurs from the node  $cn$  with the highest upper bound  $ub_{cn}$  while the remaining node  $nn$  will be explored when the algorithm backtracks to this level  $p$ . If the upper bound  $ub_{nn} \leq lb$  then the node  $nn$  will be fathomed and the algorithm backtracks to the previous level. However, when this is not the case, we can calculate an alternative upper bound  $ub'_{nn}$  as explained below.

According to the logic of our branching scheme, the latest finishing times of the branching activity  $ba$  and its predecessors  $\bar{P}_{ba}$  are decreased (or they have remained unchanged) at the left branching node while the range  $R_i$  of the other activities has remained unchanged. The opposite, however, is true for the right branching node: only the earliest finishing times of the branching activity  $ba$  and its successors  $\bar{S}_{ba}$  have possibly been increased. Therefore, we denote  $SN_x$  as the set of nodes in the network (activities) at node  $x$  (either  $cn$  or  $nn$ ) for which the range  $R_i$  has possibly changed in comparison with its parent node and the set  $SN'_x$  as the set of all other activities. Notice that  $SN_x = \{ba, \bar{P}_{ba}\}$  and  $SN'_x = M \setminus SN_x$  at the left branching node or  $SN_x = \{ba, \bar{S}_{ba}\}$  and  $SN'_x = M \setminus SN_x$  at the right branching node, as shown in Fig. 8. When the algorithm backtracks to a node  $nn$  (and consequently, node  $cn$  is already fully explored), an alternative upper bound can be calculated as follows:

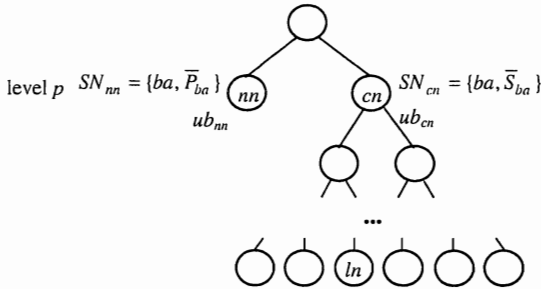
*At each new node  $nn$  for which its twin node  $cn$  is already fully explored, an alternative upper bound  $ub'_{nn}$  can be calculated as the sum of the following elements:*

- (i) *the maximal net present value for all activities  $i \in SN_{cn}$  within its range  $R_i$  at node  $nn$ ,*
- (ii) *the net present value for each activity  $i \in SN'_{cn}$  as scheduled at node  $ln$ .*

Since both parts (i) and (ii) are upper bounds for the corresponding activities, the sum  $ub'_{nn}$  is certainly an upper bound for the project at node  $nn$ . Both rules will be illustrated by means of an example in section 5. A proof is shown in the sequel of this section.

*Proof:*

In order to prove that  $ub'_{nn}$  is an alternative upper bound it is sufficient to show that both (i) and (ii) are upper bounds for the corresponding activities. Therefore, consider Fig. 8 in which the current node  $cn$  refers to the right branching node and the new node  $nn$  refers to the left branching node and consequently,  $SN_{cn} = \{ba, \bar{S}_{ba}\}$  and  $SN'_{cn} = M \setminus SN_{cn}$  (the case in which  $cn$  refers to the left branching node and  $nn$  refers to the right branching node is analogous).



**Fig. 8.** The current node  $cn$  and the new node  $nn$  at level  $p$  of the binary search tree

- (i) The upper bound calculation of condition (i) is obvious.

It simply calculates an upper bound for the net present value for all activities  $i \in SN_{cn}$  as the sum of the maximal values of each activity's profit curves within its range  $R_i = (ef_i ; lf_i)$  at node  $nn$ , i.e.  $\sum_{i \in SN_{cn}} \max_{t=ef_i, \dots, lf_i} apc_{it}$ .

- (ii) The proof of the second condition (ii) is somewhat more refined. We therefore use  $i$  to denote an activity of  $SN'_{cn}$ ,  $j$  to denote an activity of  $SN_{cn} \setminus ba$  and  $ba$  to denote the branching activity. We also use a superscript  $cn$  or  $nn$  to refer to the node in the tree (e.g.  $ef_i^{cn}$  denotes the earliest finishing time at node  $cn$  of an activity  $i \in SN'_{cn}$ ). Finally, let  $of_i^{cn}$  and  $of_i^{nn}$  be the optimal finishing times of an activity  $i$  found at the best leaf node by exploring node  $cn$  and  $nn$ , respectively (notice that we have denoted  $ln$  as the best leaf node rooted at node  $cn$ ). The proof is based on following observations:

[1] Each range  $R_i$  at node  $nn$  is a subset of the ranges of these activities at node  $cn$ , i.e.  $ef_i^{nn} = ef_i^{cn}$  and  $lf_i^{nn} \leq lf_i^{cn}$ .

[2] Each range  $R_j$  at node  $nn$  is a superset of the ranges of these activities at node  $cn$ , i.e.  $ef_j^{nn} \leq ef_j^{cn}$  and  $lf_j^{nn} = lf_j^{cn}$ .

Of course, each activity has to finish within its range. This means for activity  $i$  at node  $nn$  that  $ef_i^{nn} \leq of_i^{nn} \leq lf_i^{nn}$ . From [1] we know that  $ef_i^{nn} = ef_i^{cn}$  and  $lf_i^{nn} \leq lf_i^{cn}$  which implies that  $ef_i^{cn} \leq of_i^{nn} \leq lf_i^{cn}$ . This is exactly the range  $R_i$  of activity  $i$  at node  $cn$  in which activity  $i$  finishes at  $of_i^{cn}$ . Therefore, it will never be beneficial to have  $of_i^{nn} > of_i^{cn}$  (otherwise activity  $i$  would have finished at  $of_i^{nn}$  at node  $cn$ , i.e.  $of_i^{nn} = of_i^{cn}$ ). It is, however, perfectly possible that  $of_i^{nn} < of_i^{cn}$ , since we know from [2] that  $ef_j^{nn} \leq ef_j^{cn}$  and therefore activity  $j$  can be scheduled such that  $ef_j^{nn} \leq of_j^{nn} < ef_j^{cn}$ , eventually resulting in a higher  $npv$  for the activities  $j$ , i.e.  $\sum_j apc_{j, of_j^{nn}} > \sum_j apc_{j, of_j^{cn}}$ . In this case,

activity  $i$  is forced to finish earlier than  $of_i^{cn}$  due to a successor activity  $j$  (either direct or indirect via another activity  $i'$  of  $SN'_{cn}$ ). This means that  $of_i^{nn} + d_j = of_j^{nn}$  and  $of_j^{nn} < of_j^{cn}$ . Now we have that  $of_i^{nn} + d_j < of_j^{cn}$ . This means that the finishing times  $of_i^{nn}$  of all activities  $i$  could also have been found by exploring  $cn$  (in fact, this would have been the case if  $\sum_i apc_{i, of_i^{nn}} > \sum_i apc_{i, of_i^{cn}}$ ). Therefore it is never possible to schedule the activities  $i$  at  $of_i^{nn} < of_i^{cn}$  such that  $\sum_i apc_{i, of_i^{nn}} > \sum_i apc_{i, of_i^{cn}}$ .

It follows that the  $npv$  at the best leaf node of node  $nn$  of the activities  $i \in SN'_{cn}$  (i.e.  $\sum_i apc_{i, of_i^{nn}}$ ) will always be lower than or equal to the  $npv$  of

these activities at the best leaf node of node  $cn$  (i.e. at node  $ln$ ).

From (i) and (ii) it follows that  $ub'_{nn}$  is an alternative upper bound.

*Q.E.D.*



#### 4.4 The algorithm

The detailed steps of the branch-and-bound algorithm can be written as follows:

##### STEP 1. INITIALISATION

- Let  $lb = -\infty$  be the lower bound on the net present value of the project.
- Initialize the level of the branch-and-bound tree:  $p = 0$ .
- Do for each activity  $i$ 
  - Determine the activity profit curve,
  - Apply rule 1: restrict the range  $R_i$  (using  $wetpsp_{late}$  and  $wetpsp_{early}$ ),
  - Compute the corresponding piecewise linear overestimation curve.
- Compute an upper bound  $ub$  on the  $npv$  using the adapted version of the recursive solution for the WETPSP.
- Identify the branching activity  $ba$  as the activity with maximal vertical distance  $vd_i | i \in N$ . If no such activity exists (i.e.  $vd_i = 0$  for all  $i$ ) STOP, else go to STEP 2.

##### STEP 2. SEPARATE AND BRANCH

- Increase the level of the branch-and-bound tree:  $p = p + 1$ .
- Generate two descendant nodes in the branch-and-bound tree as follows:
  - 1<sup>st</sup> node:** Update the latest finishing times and the piecewise linear overestimation curves of the branching activity  $ba$  and its predecessors. Restrict the range  $R_i$  (using  $wetpsp_{late}$ ). Calculate an upper bound  $ub_1$  on the  $npv$ . If  $ub_1 \leq lb$  go to the 2<sup>nd</sup> node. Identify the branching activity  $ba$  as the activity with maximal vertical distance  $vd_i$ . If no such activity exists, update the feasible schedule and set  $lb = ub_1$ .
  - 2<sup>nd</sup> node:** Update the earliest finishing times and the piecewise linear overestimation curves of the branching activity  $ba$  and its successors. Restrict the range  $R_i$  (using  $wetpsp_{early}$ ). Calculate an upper bound  $ub_2$  on the  $npv$ . If  $ub_2 > lb$ , identify the branching activity  $ba$  as the activity with maximal vertical distance  $vd_i$ . If no such activity exists, update the feasible schedule and set  $lb = ub_2$ .
- Select the node with the largest upper bound  $ub = \max(ub_1, ub_2)$ . If  $ub \leq lb$ , go to STEP 3.
- Store the information for the remaining node and repeat STEP 2.

##### STEP 3. BACKTRACKING

- If the level of the branch-and-bound tree  $p = 0$ , STOP.
- If both nodes at level  $p$  have been evaluated, set  $p = p - 1$  and repeat STEP 4.
- Apply rule 2 at the remaining node  $nn$  at level  $p$  as described in section 4.3: If  $ub''_{nn} = \min\{ub_{nn}, ub'_{nn}\} \leq lb$ , set  $p = p - 1$  and repeat STEP 4.
- Go to STEP 2.

## 5. A numerical example

Consider the AON project network with 12 non-dummy activities given in Fig. 9 and its corresponding net cash flow functions as shown in Table I. The duration of each activity  $i$  is denoted above the node while the number under the node represents the cash outflow  $c_i^-$ . The project deadline  $\delta_n$  amounts to 33,  $T$  equals 10 and the discount rate  $\alpha$  equals 0.015 (1.5%). We assume the profit margin  $\gamma_i$  to be equal for each activity, i.e.  $\gamma_i = 0.30$  (30%) for each activity  $i$ . Consequently, the revenue  $c_i^+$  of each activity  $i$  amounts to  $1.30 c_i^-$ .

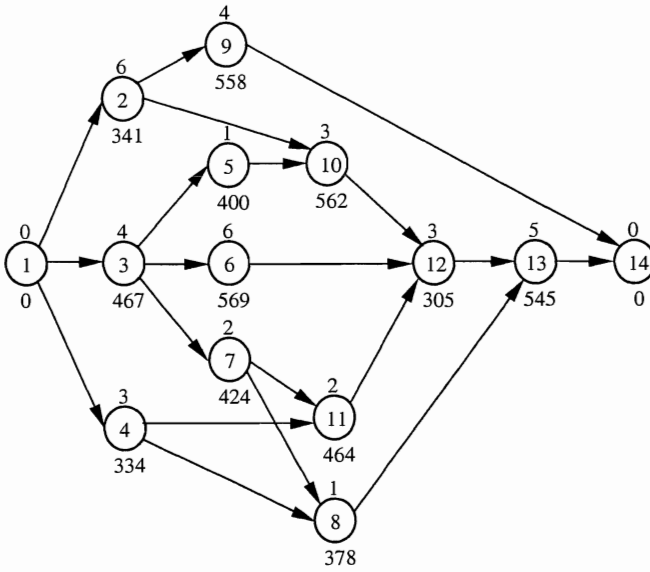


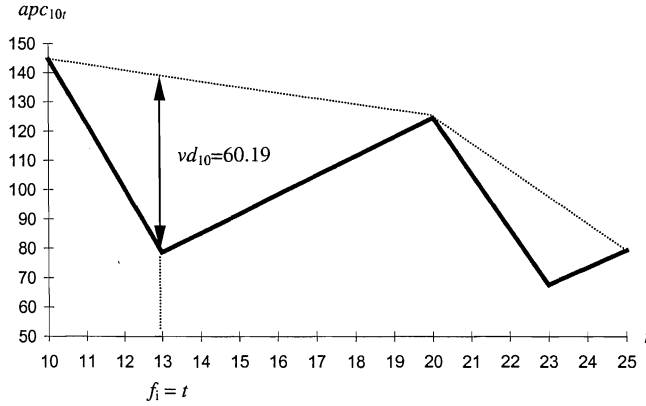
Fig. 9. An example network

The branch-and-bound tree for the example is given in Fig. 11. At the initial level  $p = 0$  of the tree, we determine the activity profit curve of each activity  $i$ . The range  $R_i = (ef_i, lf_i)$  of each activity  $i$  are calculated by simple forward and backward calculations, i.e.  $R_2 = (6, 22)$ ,  $R_3 = (4, 19)$ ,  $R_4 = (3, 23)$ ,  $R_5 = (5, 22)$ ,  $R_6 = (10, 25)$ ,  $R_7 = (6, 23)$ ,  $R_8 = (7, 28)$ ,  $R_9 = (10, 33)$ ,  $R_{10} = (9, 25)$ ,  $R_{11} = (8, 28)$ ,  $R_{12} = (13, 33)$  and  $R_{13} = (18, 33)$ . After applying rule 1, the range  $R_i$  has been restricted to  $R_2 = (6, 10)$ ,  $R_3 = (4, 10)$ ,  $R_4 = (8, 10)$ ,  $R_5 = (7, 20)$ ,  $R_6 = (10, 20)$ ,  $R_7 = (8, 20)$ ,  $R_8 = (10, 28)$ ,  $R_9 = (10, 20)$ ,  $R_{10} = (10, 25)$ ,  $R_{11} = (10, 25)$ ,  $R_{12} = (15, 28)$  and  $R_{13} = (20, 33)$ . The piecewise linear overestimation curves are determined for each activity  $i$  and an upper bound  $ub$  on the  $npv$  is computed (for an overview of the computational steps performed by the adapted recursive procedure at node 1 of the tree, we refer to the appendix). Table I gives an overview of some preliminary computations. The procedure selects activity 10 as the branching activity  $ba$  since it has the maximal vertical distance  $vd_i = 60.19$ .

**Table I.** Preliminary computations for the example network

activity	$ef_i$	$lf_i$	$e_i$	$t_{i1}$	$t_{i2}$	$h_i$	$f_i$	$vd_i$
2	6	10	4.54	0.00	0.00	10	10	0.00
3	4	10	6.31	0.00	0.00	10	8	0.00
4	8	10	4.55	0.00	0.00	10	10	0.00
5	7	20	5.36	1.44	0.00	10	10	0.00
6	10	20	0.00	2.04	0.00	10	14	22.81
7	8	20	5.64	1.52	0.00	10	10	0.00
8	10	28	0.00	1.35	2.37	10	11	52.67
9	10	20	0.00	2.00	0.00	10	14	50.94
10	10	25	0.00	2.01	9.05	10	13	<b>60.19</b>
11	10	25	0.00	1.67	7.48	10	12	57.16
12	15	28	3.75	0.00	1.91	20	20	0.00
13	20	33	0.00	1.68	7.48	20	25	35.48

Consequently, the algorithm continues with step 2. The level of the branch-and-bound tree is increased, i.e.  $p = 1$  and two descendant nodes will be generated. Therefore, the algorithm searches for the minimum point  $t = 13$  (see Fig. 10) in order to partition the activity profit curve of activity 10 into two disjoint subsets, i.e.  $R_{10} = (10,13)$  at the first node and  $R_{10} = (14,25)$  at the second node.

**Fig. 10.** Data for activity 10

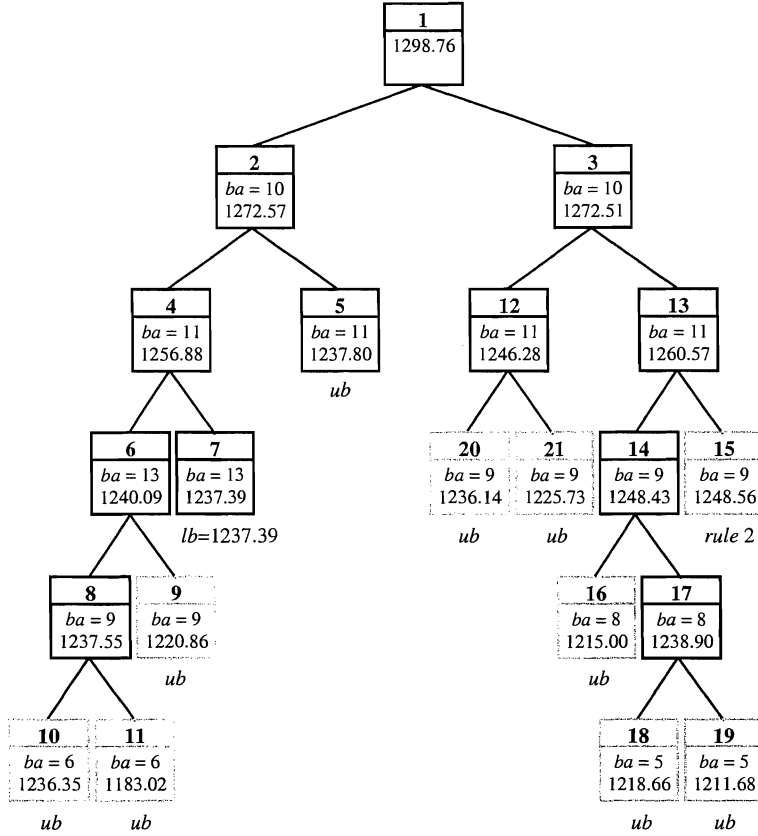
More precisely, the latest finishing times of activity 10 and its predecessors are updated at the first node on level 1 as follows (changes are marked in bold):  $R_2 = (6,10)$ ,  $R_3 = (4,9)$ ,  $R_4 = (8,10)$ ,  $R_5 = (7,10)$ ,  $R_6 = (10,20)$ ,  $R_7 = (8,20)$ ,  $R_8 = (10,28)$ ,  $R_9 = (10,20)$ ,  $R_{10} = (10,13)$ ,  $R_{11} = (10,25)$ ,  $R_{12} = (15,28)$  and  $R_{13} = (20,33)$ . Applying rule 1 by using the *wetpsp<sub>late</sub>* procedure has no effect on the range of the activities. The algorithm determines new linear piecewise overestimation curves for activities 3, 5 and 10 (since the range of the other activities has remained the same) and calculates an upper bound  $ub_1 = 1272.57$ . Since  $ub_1 > -\infty$ , we search for the branching activity *ba*

$= 11$  with the maximal vertical distance  $vd_i = 57.16$ . The range at the second node of level 1 equals  $R_2 = (6,10)$ ,  $R_3 = (4,10)$ ,  $R_4 = (8,10)$ ,  $R_5 = (7,20)$ ,  $R_6 = (10,20)$ ,  $R_7 = (8,20)$ ,  $R_8 = (10,28)$ ,  $R_9 = (10,20)$ ,  $R_{10} = (14,25)$ ,  $R_{11} = (10,25)$ ,  $R_{12} = (17,28)$  and  $R_{13} = (22,33)$ . The algorithm calculates a new upper bound  $ub_2 = 1272.51$  with the new linear piecewise overestimation curves for activities 10, 12 and 13. Since  $ub_2 > -\infty$ , the algorithm searches the branching activity  $ba$ , which is the same as in the first node, i.e.  $ba = 11$  with the maximal vertical distance  $vd_i = 57.16$ . The algorithm selects the first node with the largest upper bound  $ub = \max(1272.57, 1272.51) = 1272.57$ , stores the information of the second node and repeats step 2. Whenever the algorithm reaches a node for which no branching activity  $ba$  can be found, the lower bound  $lb$  is updated. This happens at node 7, in which the  $lb = 1237.39$ . Backtracking occurs when the upper bound  $ub$  at the second step is lower than or equal to the lower bound  $lb$  (denoted by  $ub$  in Fig. 11). This happens for the first time during the exploration of the branch-and-bound tree at node 11. The algorithm continues this way until it returns at the initial node of level 0. The exact solution of the example has a  $npv$  of 1237.39 as denoted at node 7 of the branch-and-bound tree.

Remark that node 15 with branching activity  $ba = 9$  is fathomed by the second rule of section 4.3. At this point in the search, node  $cn = 14$  is already fully explored while node  $nn = 15$  is the node we want to explore. Consequently, according to section 4.3,  $SN_{cn} = \{2,9\}$  and  $SN'_{cn} = \{1,3,4,5,6,7,8,10,11,12,13,14\}$ . The algorithm calculates an alternative upper bound at node  $nn$  as  $ub'_{nn} = \min\{1248.56, ub'_{nn}\}$ , with  $ub'_{nn}$  being the result of the following summations:

$$\begin{aligned}
 (i) \text{ at node } nn = 15, \text{ the algorithm calculates } & \sum_{i \in SN_{cn}} \max_{t=ef_i, \dots, lf_i} apc_{it} \\
 & = \max_{t=6, \dots, 10} apc_{2t} + \max_{t=15, \dots, 20} apc_{9t} = 87.79 + 123.72 = \mathbf{211.51} \\
 (ii) \text{ at node } ln = 18, \text{ the finishing times of the activities of } SA' \text{ amount to } & f_1 = 0, f_3 = 9, f_4 = 10, f_5 = 10, f_6 = 15, f_7 = 11, f_8 = 20, f_{10} = 20, f_{11} = 20, f_{12} = 25, f_{13} = 30 \\
 & \text{ and } f_{14} = 33. \text{ Node 18 is the best leaf node from the subtree rooted at the} \\
 & \text{ current node } cn = 14. \text{ Therefore, the algorithm calculates } \sum_{i \in SN'_{cn}} apc_{if'_i} = apc_{1,0} + \\
 & apc_{3,9} + apc_{4,10} + apc_{5,10} + apc_{6,15} + apc_{7,11} + apc_{8,20} + apc_{10,20} + apc_{11,20} + \\
 & apc_{12,25} + apc_{13,30} + apc_{14,33} = 114.19 + 86.07 + 103.28 + 136.13 + 107.79 + \\
 & 83.71 + 124.46 + 102.97 + 42.74 + 103.93 = \mathbf{1005.27}
 \end{aligned}$$

Consequently,  $ub'_{nn} = 211.51 + 1005.27 = 1216.78$  and therefore the upper bound at node  $nn$  equals  $\min\{1248.56, 1216.78\} = 1216.78$ . Since this upper bound is lower than the best lower bound  $lb = 1237.39$ , node 15 can be fathomed and the algorithm backtracks to the previous level and continues the search.



**Fig. 11.** The branch-and-bound tree

The branch-and-bound tree as shown in Fig. 11 contains only 21 nodes. Without the use of the two rules described in section 4.3, the tree would consist of 87 nodes. Using only rule 1 reduces the number of nodes to 25 while using only rule 2 results in a tree with 51 nodes. It is clear that the use of both rules allows for an effective pruning of the branch-and-bound tree.

## 6. Computational experience

In order to validate the branch-and-bound algorithm for the  $\max\text{-}npv^{pp}$  problem, it has been coded in Visual C++ Version 4.0 under Windows NT 4.0 on a Dell personal computer (Pentium 550 MHz processor). Therefore, we have generated 3,600 test instances with *ProGen/Max* (Schwindt, 1995). Table II represents the parameter settings used to generate the test instances. The parameters used in the full factorial experiment are indicated in bold. These instances in activity-on-the-node format use four settings for the number of activities and three settings for the order strength *OS*. The order strength, *OS* (Mastor, 1970), is defined as the number of precedence relations (including the transitive ones) divided by the theoretical

maximum number of precedence relations  $(n(n-1)/2)$ , where  $n$  denotes the number of activities). We provided the problems with cash outflows  $c_i^-$ , profit margins  $\gamma_i$ , a discount factor  $\alpha$  and a deadline  $\delta_n$ . The cash outflows were randomly generated from the interval  $[1,500]$ . Deadlines are generated by augmenting the critical path length by the numbers given in Table II. Using 10 instances for each problem class, we obtain a problem set with 3,600 test instances.

**Table II.** Parameter settings used to generate the test instances for the  $\max\text{-}npv^{pp}$

<b>Number of activities</b>	10, 20, 30, 40 or 50
Activity durations	randomly selected from the interval $[1,10]$
Number of initial and terminal activities	randomly selected from the interval $[2,4]$
Maximal number of successors and predecessors	4
<b>Order strength <math>OS</math></b> (Mastor, 1970)	0.25, 0.50 or 0.75
<b>Time period <math>T</math></b>	10, 20, 30 or 40
Discount rate $\alpha$ (in %)	0.01
Cash outflow $c_i^-$	randomly selected from the interval $[1,500]$
Profit margin $\gamma_i$	randomly selected from the interval $[5\%,30\%]$
<b>Deadline of the project <math>\delta_n</math></b>	$cpm + 0, 5, 10, 15, 20$ or $25$

Table III represents the average CPU-time and its standard deviation in seconds for a varying number of activities. Clearly, the number of activities has a significant effect on the average CPU-time. However, even problems with up to 50 activities can be solved within 0.61 seconds.

**Table III.** Impact of the number of activities

# activities	# problems	Average CPU-time	Standard Deviation
10	720	0.001	0.002
20	720	0.005	0.008
30	720	0.035	0.058
40	720	0.158	0.326
50	720	0.610	1.484

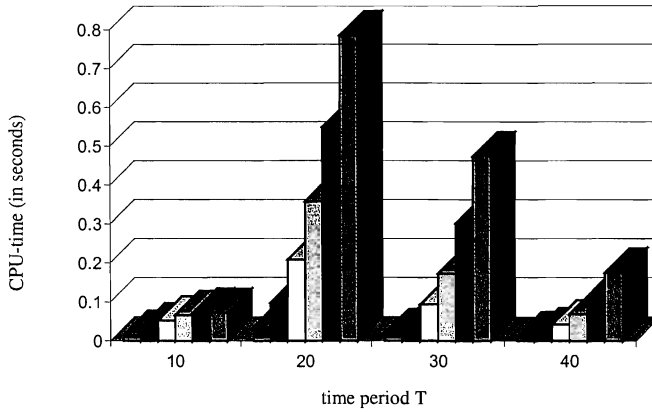
Table IV shows a negative correlation between the  $OS$  of a project and the required CPU-time, i.e. the more dense the network, the easier the problem.

**Table IV.** Impact of the order strength

$OS$	# problems	Average CPU-time	Standard Deviation
0.25	1,200	0.206	0.900
0.50	1,200	0.194	0.793
0.75	1,200	0.085	0.315

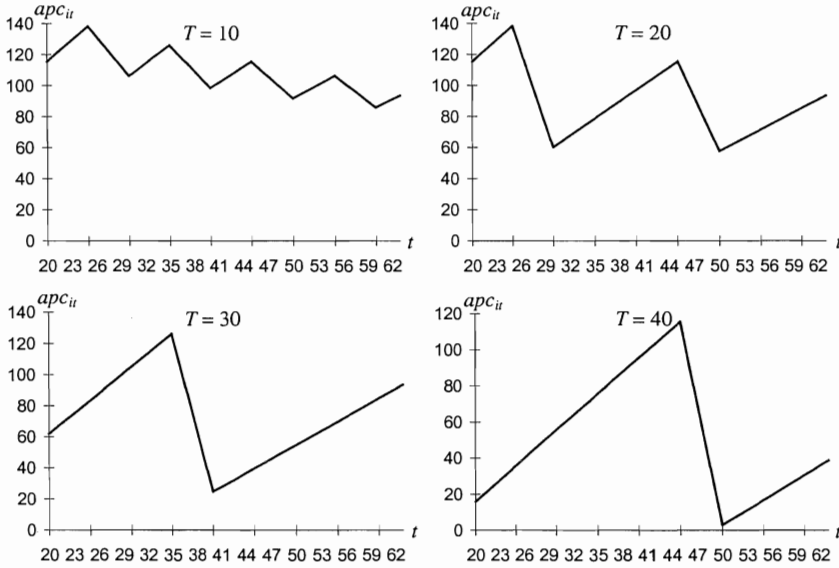
Fig. 12 displays the impact of the time period  $T$  and the deadline  $\delta_n$  on the CPU-time. The first bar of each setting for the time period  $T$  represents a deadline  $\delta_n = cpm + 0$ . The second bar denotes a deadline  $\delta_n = cpm + 5$  and so on. It is clear that the higher the deadline  $\delta_n$  of the project, the more difficult the problem becomes. This results from the fact that a large deadline  $\delta_n$  results in a large range  $R_i$  of each

activity's profit curve and consequently, a large number of nodes in the branch-and-bound tree.



**Fig. 12.** Impact of the time period  $T$  and the deadline  $\delta_i$  on the CPU-time

The impact of the time period  $T$  is somewhat different. Since the activity durations are randomly selected from the interval  $[1,10]$ , instances with a time period  $T = 10$  are rather easy to solve. This results from the fact that the quality of the overestimation curves of many activities (measured as the distance between the activity profit curve and its overestimation curve) is rather good, as shown in Fig. 13. Consequently, this results in a good upper bound  $ub$  at each node of the tree. If, however, we increase the time period  $T$ , the quality of the overestimation curve deteriorates (compare the steepness of the line segments in Fig. 13 for  $T = 10$  with  $T = 20$ ). This results in a worse upper bound  $ub$  and consequently a larger number of nodes in the branch-and-bound tree. If we increase the period  $T$  even further, problems become simpler due to the following reason: although the steepness of the line segments (compare Fig. 13 for  $T = 30$  with  $T = 40$ ) increases, the probability that the overestimation curve equals the actual activity profit curve increases. In this situation, no branching is needed and a feasible solution will be found very easily.



**Fig. 13.** The activity profit curve of an activity  $i$  with  $ef_i = 20$  and  $lf_i = 63$  with  $T = 10$ ,  $T = 20$ ,  $T = 30$  and  $T = 40$

## 7. Conclusions

This paper reports on an exact branch-and-bound procedure for problem  $cpm, \delta_n, c_j^{pp} | npv$ , i.e. the unconstrained project scheduling problem with discounted cash flows, where the net cash flows are assumed to be dependent on the completion times of the corresponding activities. Cash outflows occur when an activity is completed whereas cash inflows are incurred as progress payment at the end of some time period. The objective is to schedule the activities in order to maximize the net present value ( $npv$ ) subject to the precedence constraints and a fixed deadline.

The new branch-and-bound procedure computes upper bounds by making piecewise linear overestimations of the so-called activity profit curves. In doing so, the problem is transformed into a weighted earliness-tardiness project scheduling problem. Branching is done by partitioning the so-called branching activity, i.e. the activity with largest vertical distance between the real net present value and the value on the overestimation curve, into two disjoint subsets. Two new rules are used for node fathoming. The first rule reduces the range of each activity while the second rule calculates an alternative upper bound for each node.

Computational results revealed that problems with up to 50 activities can be solved in a very small time limit. In addition, it appears that the complexity of the  $\max-npv^{pp}$  problem is positively correlated with the project deadline. The impact of



the time period  $T$  is somewhat more confounding: problem instances with both small and large values for the time period are rather easy to solve. Problem instances with values in between appear to be more difficult.

## References

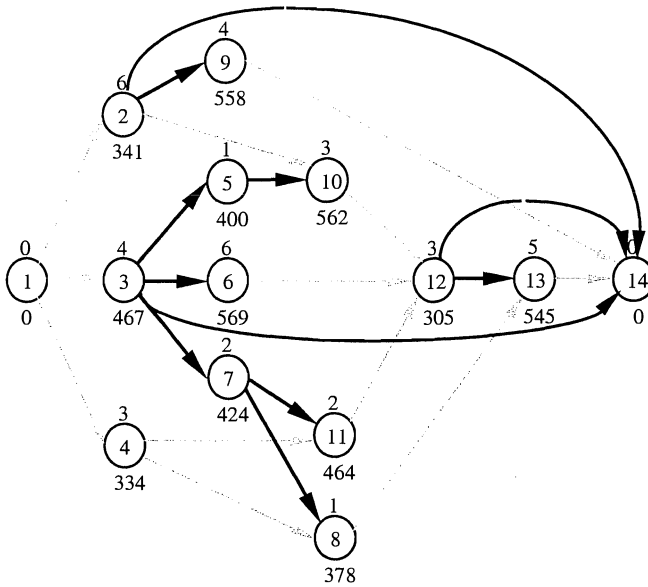
- Baroum, S.M., 1992, "An exact solution procedure for maximizing the net present value of resource-constrained projects", unpublished Ph.D. Dissertation, Indiana University.
- Baroum, S.M. and Patterson, J.H., 1996, "The development of cash flow weight procedures for maximizing the net present value of a project", *Journal of Operations Management*, 14, 209 - 227.
- Baroum, S.M. and Patterson, J.H., 1999, "An exact solution procedure for maximizing the net present value of cash flows in a network", , in: Weglarz J. (Ed.), *Handbook on Recent Advances in Project Scheduling*, Kluwer Academic Publishers, Chapter 5, 107-134.
- Dayanand, N. and Padman, R., 1993a, "The payment scheduling problem in project networks", Working Paper 9331, The Heinz School, CMU, Pittsburgh, PA 15213
- Dayanand, N. and Padman, R., 1993b, "Payments in projects: a constructor's model", Working Paper 9371, The Heinz School, CMU, Pittsburgh, PA 15213
- Dayanand, N. and Padman, R., 1997, "On modeling payments in project networks", *Journal of the Operational Research Society*, 48, 906-918.
- Doersch, R.H. and Patterson, J.H., 1977, "Scheduling a project to maximize its present value: A zero-one programming approach", *Management Science*, 23, 882-889.
- Elmaghraby, S.E. and Herroelen, W., 1990, "The scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, 49, 35-49.
- Etgar, R., Shtub, A. and LeBlanc, L.J., 1996, "Scheduling projects to maximize net present value - the case of time-dependent, contingent cash flows", *European Journal of Operational Research*, 96, 90-96.
- Etgar, R. and Shtub, A., 1999, "Scheduling project activities to maximize the net present value - the case of linear time dependent, contingent cash flows", *International Journal of Production Research*, 37, 329-339.
- Grinold, R.C., 1972, "The payment scheduling problem", *Naval Research Logistics Quarterly*, 19, 123-136.
- Herroelen, W. and Gallens, E., 1993, "Computational experience with an optimal procedure for the scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, 65, 274-277.
- Herroelen, W., Demeulemeester, E. and Van Dommelen, P., 1997, "Project network models with discounted cash flows: A guided tour through recent developments", *European Journal of Operational Research*, 100, 97-121.
- Herroelen, W., Demeulemeester, E. and De Reyck, B., 1999, "A classification scheme for project scheduling problems", in: Weglarz J. (Ed.), *Handbook on Recent Advances in Project Scheduling*, Kluwer Academic Publishers, Chapter 1, 1-26.
- Icmeli, O. and Erengüç, S.S., 1994, "A tabu search procedure for resource-constrained project scheduling with discounted cash flows", *Computers and Operations Research*, 21, 841-853.

- Icmeli, O. and Erengüç, S.S., 1996, "A branch-and-bound procedure for the resource-constrained project scheduling problem with discounted cash flows", *Management Science*, 42, 1395-1408.
- Kazaz, B. and Sepil, C., B., 1996, "Project scheduling with discounted cash flows and progress payments", *Journal of the Operational Research Society*, 47, 1262-1272.
- Özdamar, L., Ulusoy, G. and Bayyigit, M., 1994, "A heuristic treatment of tardiness and net present value criteria in resource-constrained project scheduling", Working Paper, Department of Industrial Engineering, Marmara University.
- Padman, R., Smith-Daniels, D.E. and Smith-Daniels, V.L., 1997, "Heuristic scheduling of resource-constrained projects with cash flows", *Naval Research Logistics*, 44, 365-381.
- Padman, R. and Smith-Daniels, D.E., 1993, "Early-tardy cost trade-offs in resource constrained projects with cash flows: An optimization-guided heuristic approach", *European Journal of Operational Research*, 64, 295-311.
- Patterson, J.H., Slowinski, R., Talbot, F.B. and Weglarz, J., 1989, "An algorithm for a general class of precedence and resource constrained scheduling problems", Part I, Chapter 1 in Slowinski, R. & Weglarz, J. (eds.), *Advances in Project Scheduling*, Elsevier Science Publishers, Amsterdam, 3-28.
- Patterson, J.H., Talbot, F.B., Slowinski, R. and Weglarz, J., 1990, "Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems", *European Journal of Operational Research*, 49, 68-79.
- Pinder, J.P. and Maruchek, A.S., 1996, "Using discounted cash flow heuristics to improve project net present value", *Journal of Operations Management*, 14, 229-240.
- Russell, A.H., 1970, "Cash flows in networks", *Management Science*, 16, 357-373.
- Russell, R.A., 1986, "A comparison of heuristics for scheduling projects with cash flows and resource restrictions", *Management Science*, 32, 291-300.
- Sepil, C. and Ortaç, N., 1997, "Performance of the heuristic procedures for constrained projects with progress payments", *Journal of the Operational Research Society*, 48, 1123-1130.
- Shtub, A. and Etgar, R., 1997, "A branch-and-bound algorithm for scheduling projects to maximize net present value: the case of time dependent, contingent cash flows", *International Journal of Production Research*, 35, 3367-3378.
- Smith-Daniels, D.E. and Aquilano, N.J., 1987, "Using a late-start resource-constrained project schedule to improve project net present value", *Decision Sciences*, 18, 617-630.
- Smith-Daniels, D.E. and Smith-Daniels, V.L., 1987, "Maximizing the net present value of a project subject to materials and capital constraints", *Journal of Operations Management*, 7, 33-45.
- Smith-Daniels, D.E., Padman, R. and Smith-Daniels, V.L., 1996, "Heuristic scheduling of capital constrained projects", *Journal of Operations Management*, 14, 241-254.
- Ulusoy, G. and Özdamar, L., 1995, "A heuristic scheduling algorithm for improving the duration and net present value of a project", *International Journal of Operations and Production Management*, 15, 89-98.
- Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2000, "An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem", to appear in *Annals of Operations Research*.

- Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 1999a, "On maximizing the net present value of a project under resource constraints", Research Report 9915, Department of Applied Economics, Katholieke Universiteit Leuven.
- Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 1999b, Scheduling projects with linear time-dependent cash flows to maximize the net present value, Research Report 9949, Department of Applied Economics, Katholieke Universiteit Leuven.
- Yang, K.K., Talbot, F.B. and Patterson, J.H., 1992, "Scheduling a project to maximize its net present value: An integer programming approach", *European Journal of Operational Research*, 64, 188-198.
- Yang, K.K., Tay, L.C. and Sum, C.C., 1995, "A comparison of stochastic scheduling rules for maximizing project net present value", *European Journal of Operational Research*, 85, 327-339.
- Zhu, D. and Padman, R., 1996, "A tabu search approach for scheduling resource-constrained projects with cash flows", Working Paper 96-30, Carnegie-Mellon University.
- Zhu, D. and Padman, R., 1997, "Connectionist approaches for solver selection in constrained project scheduling", *Annals of Operations Research*, 72, 265-298.

## APPENDIX

In this appendix we give a detailed description of the computational steps performed by the adapted recursive algorithm at node 1 of the branch-and-bound tree of Fig. 11. In Fig. 14, we show the due date tree  $DT$  with finishing times  $f_1 = 0, f_2 = 10, f_3 = 10, f_4 = 10, f_5 = 11, f_6 = 16, f_7 = 12, f_8 = 13, f_9 = 14, f_{10} = 14, f_{11} = 14, f_{12} = 20, f_{13} = 25$  and  $f_{14} = 33$ . We now continue with the second step of the recursive search procedure, as described by Vanhoucke et al. (2000). Remark that  $ET$  denotes the earliness-tardiness cost and  $CA$  the set of already considered activities.



**Fig. 14.** The due date tree  $DT$  of the example network in Fig. 9 at the node 1 of the branch-and-bound tree of Fig. 11

### STEP2:

#### RECURSION(14)

$SA = \{14\}, CA = \{14\}, ET = 0.00$

#### RECURSION(3)

$SA = \{3\}, CA = \{3, 14\}, ET = 6.31$

#### RECURSION(5)

$SA = \{5\}, CA = \{3, 5, 14\}, ET = -1.44$

#### RECURSION(10)

$SA = \{10\}, CA = \{3, 5, 10, 14\}, ET = -2.01$

$ET' = -2.01, SA = \{5, 10\}, ET = -3.45$

$ET' = -3.45, SA = \{3, 5, 10\}, ET = 2.86$

**RECURSION(6)**

$SA = \{6\}$ ,  $CA = \{3,5,6,10,14\}$ ,  $ET = -2.04$   
 $ET' = -2.04$ ,  $SA = \{3,5,6,10\}$ ,  $ET = 0.82$

**RECURSION(7)**

$SA = \{7\}$ ,  $CA = \{3,5,6,7,10,14\}$ ,  $ET = -1.52$

**RECURSION(8)**

$SA = \{8\}$ ,  $CA = \{3,5,6,7,8,10,14\}$ ,  $ET = -1.35$   
 $ET' = -1.35$ ,  $SA = \{7,8\}$ ,  $ET = -2.88$

**RECURSION(11)**

$SA = \{11\}$ ,  $CA = \{3,5,6,7,8,10,11,14\}$ ,  $ET = -1.67$   
 $ET' = -1.67$ ,  $SA = \{7,8,11\}$ ,  $ET = -4.54$   
 $ET' = -4.54$ ,  $SA = \{3,5,6,7,8,10,11\}$ ,  $ET = -3.73$   
 $\min\{v, w, w', w''\} = \min\{1, 1, \infty, 3\} = 1$   
 Decrease the activity completion times  $i \in SA$ :  $f_3 = 9, f_5 = 10, f_6 = 15, f_7 = 11$ ,  
 $f_8 = 12, f_{10} = 13$  and  $f_{11} = 13$

**Step2:****RECURSION(14)**

$SA = \{14\}$ ,  $CA = \{14\}$ ,  $ET = 0.00$

**RECURSION(3)**

$SA = \{3\}$ ,  $CA = \{3,14\}$ ,  $ET = 6.31$

**RECURSION(5)**

$SA = \{5\}$ ,  $CA = \{3,5,14\}$ ,  $ET = 5.36$

**RECURSION(10)**

$SA = \{10\}$ ,  $CA = \{3,5,10,14\}$ ,  $ET = -2.01$   
 $ET' = -2.01$ ,  $SA = \{5,10\}$ ,  $ET = 3.35$   
 Insert arc (5,14) in the due date tree  
 Delete arc (3,5) from the due date tree

**RECURSION(6)**

$SA = \{6\}$ ,  $CA = \{3,5,6,10,14\}$ ,  $ET = -2.04$   
 $ET' = -2.04$ ,  $SA = \{3,6\}$ ,  $ET = 4.27$

**RECURSION(7)**

$SA = \{7\}$ ,  $CA = \{3,5,6,7,10,14\}$ ,  $ET = -1.52$

**RECURSION(8)**

$SA = \{8\}$ ,  $CA = \{3,5,6,7,8,10,14\}$ ,  $ET = -1.35$   
 $ET' = -1.35$ ,  $SA = \{7,8\}$ ,  $ET = -2.88$

**RECURSION(11)**

$SA = \{11\}$ ,  $CA = \{3,5,6,7,8,10,11,14\}$ ,  $ET = -1.67$   
 $ET' = -1.67$ ,  $SA = \{7,8,11\}$ ,  $ET = -4.54$   
 $ET' = -4.54$ ,  $SA = \{3,6,7,8,11\}$ ,  $ET = -0.27$   
 $\min\{v, w, w', w''\} = \min\{1, 1, \infty, 2\} = 1$   
 Decrease the activity completion times  $i \in SA$ :  $f_3 = 8, f_6 = 14, f_7 = 10, f_8 = 11$   
 and  $f_{11} = 12$

**Step2:****RECURSION(14)**

$SA = \{14\}$ ,  $CA = \{14\}$ ,  $ET = 0.00$

**RECURSION(3)**

$SA = \{3\}, CA = \{3,14\}, ET = 6.31$

**RECURSION(6)**

$SA = \{6\}, CA = \{3,6,14\}, ET = -2.04$

$ET' = -2.04, SA = \{3,6\}, ET = 4.27$

**RECURSION(7)**

$SA = \{7\}, CA = \{3,6,7,14\}, ET = 5.64$

**RECURSION(8)**

$SA = \{8\}, CA = \{3,6,7,8,14\}, ET = -1.35$

$ET' = -1.35, SA = \{7,8\}, ET = 4.29$

**RECURSION(11)**

$SA = \{11\}, CA = \{3,6,7,8,11,14\}, ET = -1.67$

$ET' = -1.67, SA = \{7,8,11\}, ET = 2.62$

Insert arc (7,14) in the due date tree

Delete arc (3,7) from the due date tree

$ET' = 4.27, SA = \{3,6,14\}, ET = 4.27$

**RECURSION(12)**

$SA = \{12\}, CA = \{3,6,12,14\}, ET = 3.57$

**RECURSION(13)**

$SA = \{13\}, CA = \{3,6,12,13,14\}, ET = -1.68$

$ET' = -1.68, SA = \{12,13\}, ET = 1.89$

$ET' = 1.89, SA = \{3,6,12,13,14\}, ET = 6.16$

**RECURSION(2)**

$SA = \{2\}, CA = \{2,3,6,12,13,14\}, ET = 4.54$

**RECURSION(9)**

$SA = \{9\}, CA = \{2,3,6,9,12,13,14\}, ET = -2.00$

$ET' = -2.00, SA = \{2,9\}, ET = 2.54$

$ET' = 2.54, SA = \{2,3,6,9,12,13,14\}, ET = 8.70$

The recursive algorithm stops and reports the finishing times  $f_1 = 0, f_2 = 10, f_3 = 8, f_4 = 10, f_5 = 10, f_6 = 14, f_7 = 10, f_8 = 11, f_9 = 14, f_{10} = 13, f_{11} = 12, f_{12} = 20, f_{13} = 25$  and  $f_{14} = 33$  as shown in Table I.